JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 13, No. 1, January 2016

Intelligent Motion Video Guidance for Unmanned Air System Ground Target Surveillance

John Valasek,* Kenton Kirkpatrick,† James May,‡ and Joshua Harris§
Texas A&M University, College Station, Texas 77843-3141

DOI: 10.2514/1.I010198

Unmanned air systems with video capturing systems for surveillance and visual tracking of ground targets have worked relatively well when employing gimbaled cameras controlled by two or more operators: one to fly the vehicle, and one to orient the camera and visually track ground targets. However, autonomous operation to reduce operator workload and crew levels is more challenging when the camera is strapdown, or fixed to the airframe without a pan-and-tilt capability, rather than gimbaled, so that the vehicle must be steered to orient the camera field of view. Visual tracking becomes even more difficult when the target follows an unpredictable path. This paper investigates a machine learning algorithm for visual tracking of stationary and moving ground targets by unmanned air systems with nongimbaling, fixed pan-and-tilt cameras. The algorithm is based on *Q* learning, and the learning agent initially determines an offline control policy for vehicle orientation and flight path such that a target can be tracked in the image frame of the camera without the need for operator input. Performance of the control policy is demonstrated with simulation test case scenarios for stationary, linear, and random moving targets with changes in target speeds and trajectories. Monte Carlo results presented in the paper demonstrate that the learned policies are capable of tracking stationary and moving targets with path perturbations, provided the perturbations are small. The learned policies are robust to small changes in target trajectory; therefore, learning separate policies for every type of trajectory is not required. The approach is judged to have merit for autonomous visual tracking of both fixed and randomly moving ground targets.

I. Introduction

TACTICAL unmanned air system (UAS) intelligence, surveillance, and reconnaissance missions often require four crewmen: an air vehicle operator, a payload vehicle operator, a mission commander, and an intelligence "gatherer" (Fig. 1). Not only does this require significant staffing levels, but it can lead to inefficiency of operations. The air vehicle operator is responsible for flying the vehicle; controlling its trajectory for tracking; and ensuring smooth, stable flight for obtaining quality video data. The payload vehicle operator views video camera output and communicates verbally with the air vehicle operator to provide trajectory commands for functions such as maintaining tracking on a current target, switching to another target, flying in a specified area, conducting a search in a specified area, etc. This arrangement requires a high level of coordination between the two operators and, in spite of training for various operational events, problems can occur. For instance, if the air vehicle operator flies directly over a target, the image from a gimbaled camera is disorienting to the payload vehicle operator and the commander and gatherer, since the image field of view (FOV) will appear to advance to the target and then suddenly retreat away from the target in the opposite direction. The problem is even more challenging if the camera orientation is strapdown or fixed to the airframe without a pan-and-tilt capability rather than gimbaled. This is the case on UASs that are unable to carry a gimbaled camera, where the only way to track targets is to steer the UAS to orient the FOV of the camera.

A motion video system capable of simultaneously controlling a UAS and its camera in order to keep selected targets visible in the FOV would reduce the number of crew by at least one, and it would free the remaining crew to focus on selecting viable targets, analyzing the images received, and conducting surveillance. One approach is to determine a control policy that is capable of controlling the UAS autonomously along a certain trajectory while a human operates the camera. Another approach is to do the opposite, in which the UAS is flown manually while the camera autonomously gimbals to track identified targets. Introducing autonomy to the UAS motion video tracking problem allows a third approach, in which the UAS autonomously flies and autonomously tracks targets, potentially leading to greater flight and tracking efficiency.

Motion video tracking of targets with fixed-wing UASs using both gimbaled and nongimbaled cameras has been addressed in the literature, with the gimbaled approaches being more numerous. Considering the class of gimbaled camera approaches to target tracking with path-planning solutions, the general approach is to estimate the position of the aircraft, the position of the target, and the wind based on current measurements, as well as to use the current estimate to regenerate an open-loop trajectory. Rysdyk developed flight-path geometry, guidance laws, and synchronous camera angles to observe a static ground target [1,2]. Analytic expressions for paths that result in constant line-of-sight orientation of the target relative to the aircraft body frame were developed and, using minimal heuristics, a guidance law based on "good helmsman" behavior was developed and implemented that resulted in constant line-of-sight orientation relative to the vehicle. Stolle and Rysdyk used dynamic planning to track a moving target by defining maneuvers that resulted in maximum target exposure, and they provided guidance laws to allow path following for these maneuvers [3]. The complete path about the target at constant altitude and airspeed was specified simply by the desired orbital segment and course reversals at the segment boundaries. The UAS was equipped with a nose-mounted camera capable of pan-and-tilt rotation with angles adjusted automatically in order to keep the target in the camera's field of view.

Presented as Paper 2012 at the 2012 AIAA Infotech@Aerospace Conference, Garden Grove, CA, 20–22 June 2012; received 14 October 2013; revision received 20 July 2015; accepted for publication 14 September 2015; published online 28 December 2015.

*Professor and Director, Vehicle Systems and Control Laboratory, Aerospace Engineering Department; valasek@tamu.edu. Associate Fellow AIAA.
†Postdoctoral Research Associate, Vehicle Systems and Control Laboratory, Aerospace Engineering Department; kentonkirk@gmail.com. Member AIAA.
‡Graduate Research Assistant, Vehicle Systems and Control Laboratory, Aerospace Engineering Department; jim.may.jr@gmail.com. Student Member AIAA.
§Graduate Research Assistant, Vehicle Systems and Control Laboratory, Aerospace Engineering Department; joshua.a.harris@tamu.edu. Student Member AIAA.

Fig. 1 Typical UAS crew assets for intelligence, surveillance, and reconnaissance operations.

An algorithm based on a waypoint strategy was created by Lee et al. [4]. Using this navigation scheme, the aircraft flies in a sinusoidal manner and changes the amplitude of the sinusoid, all while maintaining a constant velocity and tracking of a ground target that has varying speed. The algorithm also allows the aircraft to track the ground target with an offset vector (e.g., the user may wish the aircraft to stay ahead of the ground target or to its sides). Rafi et al. developed a single circular pattern navigation algorithm for targets moving at any speed with any pattern, without switching between different navigation strategies in different scenarios [5]. The camera was controlled by the algorithm according to the position and orientation of the aircraft and the position of the target. Quigly et al. created a human–UAS interaction scheme using a flight-path generation algorithm in which the operator manually flew the UAS to produce an estimation of the target position [6]. This allowed the vehicle to fly itself and control the gimbal while the operator refined and moved the target position. A path-planning-based solution with multiple vehicles has been demonstrated by Bethke et al. [7]. A cooperative vision-based estimation and tracking system was used to detect, estimate, and track the location and velocity of objects in three dimensions. The tracking algorithm exploited cooperation between multiple aircraft and allowed good tracking of the target without the need for a single vehicle to execute maneuvers to gain better vantage points. The algorithm can be distributed to multiple aircraft, and it was shown to give better results than could be achieved with a single aircraft while being robust to failures. Dobrokhodov et al. developed a feedback control law for autonomous tracking of a moving target from kinematic laws while simultaneously estimating GPS coordinates of the target. It was assumed that the target velocity was known, that there was no wind, and that the camera was gimbaled [8].

Removing the ability to gimbal introduces additional kinematic constraints, and path planning becomes more difficult. For fixed-camera visual tracking, one of the biggest challenges stems from the need to determine an optimal control policy for keeping the target in the image frame when targets move with unknown paths. Conventional control techniques require determining an appropriate cost function and then finding the weights that make the control optimal. Although finding the optimal control is often straightforward, determining the cost function that best describes the problem is not straightforward. Additionally, a sudden change in the course of the target may push it outside the FOV before a response is generated, possibly causing the aircraft to lose the target altogether. The measurements of the target location are inherently nonlinear in the single-vehicle case because the observed state variables are measured angles to the target. For all of these reasons, motion video tracking with a fixed camera is not trivial.

Redding et al. elucidated the complexity of localizing a ground-based object when imaged from a small fixed-wing UAS [9]. Using the pixel location of the target in an image, with measurements of aircraft position and attitude, as well as camera pose angles, the target was localized in inertial coordinates. Although tracking of static and moving targets was not considered, possible error sources and localization sensitivities to each source were presented, which were useful for the tracking problem. Egbert and Beard used a nongimbaled electro-optical camera in the loop to track linear features such as a border or road pathway [10,11]. This was done by determining the required altitude above the ground level constraint imposed on a bank-to-turn micro air vehicle to maintain the pathway in the footprint of the camera and the associated bank angle constraints. Stationary or moving point targets were not considered, but this work serves to highlight the complexities of the geometry and sources of error for this class of problem.

Saunders and Beard addressed the problem of tracking a ground-based target with a fixed camera that was pointing out the wing of a UAS [12]. Rather than planning explicit trajectories for the vehicle, a nonlinear image-based feedback guidance strategy was developed that maintained the target in the FOV of the camera. A-priori-determined terrain features such as roads and buildings enhanced the tracking performance. Theodorakopolous and Lacroix created and tested a two-step heading-based flight control strategy for maximizing target visibility based on proportional navigation [13]. In the first step, a near-optimal heading angle was determined that was a tradeoff between the target distance, range, and lateral tracking error of the target, which was to be minimized. In the second step, a lateral guidance algorithm was used to track the desired heading angle by computing the current heading angle and the horizontal distance to the target and then determining a roll command. A heuristic strategy was used to determine a spiraling strategy to ensure that the target distance remained below the maximal detection radius and to satisfy the minimum turning radius.

This paper builds on the work and results of [12,13] by developing a reinforcement learning (RL) algorithm for determining the optimal control policy that steers a fixed-camera aircraft to provide automated tracking of both stationary and moving ground targets. The specific RL algorithm is based on Q learning, in which the agent learns to keep a target in the camera FOV by controlling the aircraft's orientation and flight path. The algorithm determines the desired bank angle commands for successful tracking at each instant in time, and a lateral controller implements the commands. Two RL algorithms are used in this paper: Watkins Q learning, and Watkins Q learning with eligibility traces. There are three advantages to this approach. First, all learning, computation, and tracking takes place in the camera's image frame to simplify computations, and prior information about the geometry of the target space (nodes or road network, landmarks, features, coordinates) are not required to do the tracking. This permits tracking of targets in completely featureless terrain environments. Second, knowledge of the target dynamics in the image frame is not needed. Third, determining an accurate dynamical model relating the target position and motion in the image frame to aircraft states and controls can be difficult, and the model will have uncertainties resulting from the modeling process. The modeled kinematic chain from target position in the inertial frame to the aircraft frame introduces computational errors that propagate with each successive rotation. By using an RL approach, the control policy can be determined without needing to first identify the entire system dynamics. A feature of this approach is that the learning agent will continue to learn, refine, and update the control policy previously learned offline during actual operation. Use of this approach

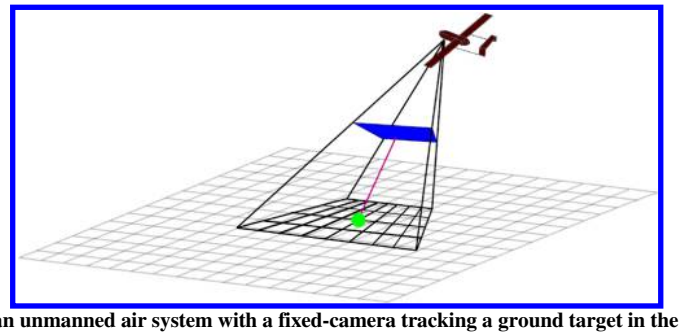

Fig. 2 Geometry of an unmanned air system with a fixed-camera tracking a ground target in the camera image frame.

does not require system models, although they can help offline learning, and the reinforcement learning eliminates the need for any heuristic or ad hoc approaches. Although the learning used in the present work is done offline, a black-box simulator of the target will not always be available for real applications. In this case, more advanced reinforcement learning techniques such as model-based RL techniques, which learn a system model from the environmental data; function approximation techniques; and hierarchal learning can be used [14–17]. These methods can result in improved performance with less sample complexity, though usually at the cost of increased computational complexity [18], which is a tradeoff well suited for offline learning. However, Q learning is used as the RL algorithm in this paper due to its simplicity and ease of implementation.

II. Tracking Problem Representation and Algorithm

Figure 2 shows the geometry of the tracking problem. The dot in the figure represents the target location on the ground, and the gridlines show how it appears in the image frame. The line from the center of the image frame to the target shows the direct line of sight. The target is assumed to already be recognized in the image frame, but lost targets are not reacquired by the algorithm. The tracking problem investigated in this work assumes a camera with a FOV of 40 deg and an aspect ratio of 4 : 3. In a separate offline learning exercise, the Watkins Q-learning algorithm was used to determine the stationary camera tilt angle of -20 deg that provides the best chance of tracking success, which is defined here as keeping the target within the camera FOV.

Reinforcement learning is a process of learning through interaction, in which a program uses previous knowledge of the results of its actions in each situation to make an informed decision when it later returns to the same situation. In RL, environments are typically modeled as a Markov decision process, so the actions selected at any given point in time depend only on the current state of the system. It is a method that has been used for many diverse problems, ranging from board games to behavior-based robotics. The purpose of the learning agent used in RL is to maximize the long-term cumulative reward, and not just the immediate reward [19]. In this work, the goal is to keep the target in the image frame, with a preference for being far from the edges. The reward structure is formulated to represent these objectives. The agent uses the knowledge gained by reward maximization to update a control policy that is a function of the states and actions. This control policy is essentially a large matrix that is composed of every possible state–action pair discretization point as its entries. There are many classes of RL algorithms, including Monte Carlo techniques, model-free techniques based on temporal difference (TD) learning, model-based techniques aided by dynamic programming, and policy search techniques. Monte Carlo methods only allow learning to occur at the end of each episode, causing problems that have long episodes to have a slow learning rate and making online learning more challenging. TD methods have the advantage of being able to learn at every time step without requiring the input of an environmental model. The Watkins Q learning and Watkins Q learning with eligibility traces are the algorithms chosen and are described in the next sections.

A. Reinforcement Learning Representation

To use RL for this problem, a representation in terms of states, actions, goals, and rewards must be designed. Referring to Fig. 3, the states of the RL agent s are defined for this problem to be those states of the system that either give information regarding the target's position or those of the aircraft that can be controlled for tracking.

This yields a state space consisting of three variables: target x position in the image X_i, target y position in the image Y_i, and aircraft bank angle ϕ:

$$\boldsymbol{s} = [X_{\mathrm{i}} \quad Y_{\mathrm{i}} \quad \phi]^{\top} \tag{1}$$

The goal for an RL agent is defined using the reward structure. The overall goal is to have the target remain in the image frame once commanded, so a proper set of reward requirements must be constructed. Since leaving the image frame is considered the worst result, it retains the worst reward. In this case, after some initial iterations, a value of $r = -20$ is given for a target leaving the image frame. It is also desired to remain away from the edges of the image, so a reward of $r = -5$ is given for hitting the image boundary, whereas a positive reward of $r = +20$ is awarded for

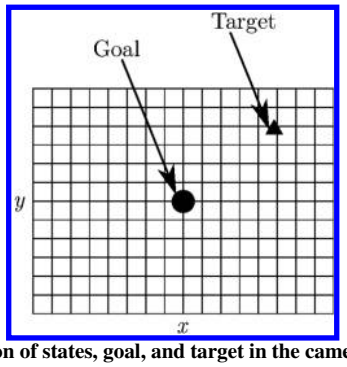

Fig. 3 Definition of states, goal, and target in the camera image frame.

reaching the center of the image. This encourages the RL agent to move the aircraft such that the target stays as far from any edge of the image frame as is possible. All other possible states yield a neutral reward of $r = 0$. Since the goal is generally defined for the RL agent as the state that yields the highest reward, the goal is defined as all states where the target image position is at the center.

In this work, the target is always unaware of the aircraft, and therefore is never able to choose its own actions in accordance with avoiding the aircraft. For each learning scenario, the target may only follow the policy assigned to it at the beginning of the episode. Thus, the agent for this problem is limited in its control of the states because the target global position is independent of any action the aircraft takes. The only part of the environment that the aircraft can control is itself. Based on the assumption for this problem that the aircraft will be traveling at constant altitude with constant cruise speed, the only way the aircraft can control the position of the target in the image frame is to change its bank angle. In a tabular representation, the resulting action-value function is a matrix composed of discrete states and actions. Since this is a continuous-state system, function approximation is required to account for states that lie between the discrete entries. For this work, the k-nearest neighbors were used for function approximation. This approach assumes that each state is closest in value to the discrete state that is located at the closest distance to it in information space. It is a simple approach that can be highly accurate as long as the state-space discretization is not too coarse.

The action space for this problem is defined to be increments in the aircraft bank angle, where for this problem, $\Delta\phi = 2$ deg:

$$\boldsymbol{a} = [-2 \quad 0 \quad 2]^{\mathsf{T}} \tag{2}$$

This representation of the RL problem is therefore

$$\boldsymbol{s} = [X_{\mathrm{i}} \quad Y_{\mathrm{i}} \quad \phi]^{\mathsf{T}} \tag{3}$$

$$\boldsymbol{a} = [-\Delta\phi \quad 0 \quad +\Delta\phi]^{\mathsf{T}} \tag{4}$$

$$\boldsymbol{g} = [X_{\mathrm{ic}} \quad Y_{\mathrm{ic}} \quad \phi]^{\mathsf{T}} \tag{5}$$

In Eq. (5), X_{ic} and Y_{ic} are the pixel coordinates for the center of the image. This represents a goal state of reaching the image center without any restriction on bank angle.

B. Watkins Q Learning

Watkins Q learning [20] is a commonly used form of TD. It is an offpolicy method that uses an action-value function update rule based on the equation [19]

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha[r_{k+1} + \gamma \max_a Q(s_{k+1}, a) - Q(s_k, a_k)] \tag{6}$$

where s_k is the current state, a_k is the current action, s_{k+1} is the next state, r_{k+1} is the next reward, Q is the action-value function (used in the control policy), and k is the time step. The parameter α is the step-size parameter that controls how fast the value function changes on a single time step. The parameter γ is the future policy discount factor. The learning rate α and discount rate γ are design parameters that are held constant. Since Q learning is offpolicy, it learns the greedy policy while choosing actions based on an exploratory policy. In this work, an ϵ-greedy policy is used that chooses exploratory actions with a probability of $\epsilon \in [0, 1]$.

C. Watkins Q Learning with Eligibility Traces

Eligibility traces are an RL parameter that add "memory" by recording how recently a state has been visited by the agent. Eligibility traces are so named because they describe how "eligible" a state is for a learning update. When a learning event occurs, the states that have been more recently visited (i.e., those with a larger eligibility trace value) receive a larger change to their action values. Eligibility traces decay over time by a parameter called the trace-decay parameter, denoted by the symbol $\lambda \in [0, 1]$.

Since Q learning is used for the nominal tracking agent, the Watkins $Q(\lambda)$ algorithm is used to incorporate eligibility traces $e(s, a)$ to the learning agent [21]. The $Q(\lambda)$ algorithm maintains the offpolicy behavior of standard Q learning. Since the agent learns the greedy policy while typically following an exploratory policy, $Q(\lambda)$ resets the eligibility traces to zero following a nongreedy action. The action-value update for $Q(\lambda)$, reproduced from [19], becomes

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha\delta_k e_k(s, a) \tag{7}$$

where

$$\delta_k = r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k) \tag{8}$$

and

$$e_k(s, a) = \delta_{ss_k}\delta_{aa_k} + \begin{cases} \gamma\lambda e_{k-1}(s, a) & \text{if } Q_{k-1}(s, a_k) = \max_a Q_{k-1}(s, a) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Equation (9) updates the eligibility traces at the current time step. In Eq. (9), δ_{ss_k} and δ_{aa_k} are the Kronecker delta, and not the TD error defined by Eq. (8). The product $\lambda\gamma$ decays the value of the eligibility trace at each time step, and the value of the eligibility trace for a state-action pair (s, a) is incremented by unity every time the agent chooses action a at state s.

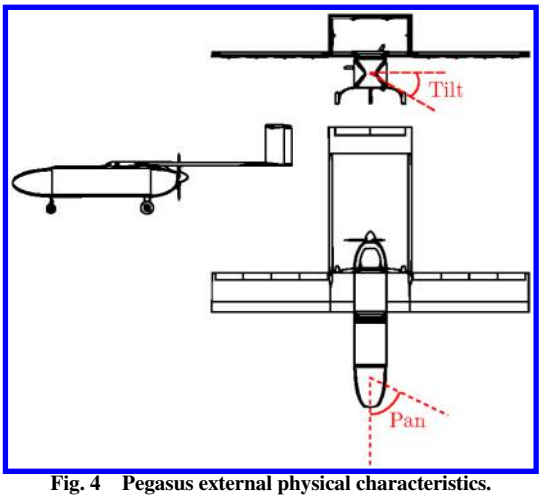

Fig. 4 Pegasus external physical characteristics.

III. Environment Modeling and Learning Simulation

The environment is modeled by converting the target and aircraft positions in the global simulation space to the target position in the image frame and the bank angle command in the learning space. The distance of the target from the global simulation space to the learning space is determined by

$$d = \frac{(\hat{\boldsymbol{x}}_{\text{ti}} - \hat{\boldsymbol{x}}_{\text{t}}) \cdot \hat{\boldsymbol{n}}_{\text{ti}}}{(\hat{\boldsymbol{x}}_{\text{c}} - \hat{\boldsymbol{x}}_{\text{t}}) \cdot \hat{\boldsymbol{n}}_{\text{ti}}} \tag{10}$$

where $\hat{\boldsymbol{x}}_{\text{t}}$ is the target inertial position vector, $\hat{\boldsymbol{x}}_{\text{c}}$ is the camera inertial position vector, $\hat{\boldsymbol{x}}_{\text{ti}}$ is the position vector of the target in the image frame with respect to inertial coordinates, and $\hat{\boldsymbol{n}}_{\text{ti}}$ is a unit vector. The kinematic model governing the aircraft motion uses parameters based on the Pegasus UAS (Fig. 4). The kinematic model has position states (x, y, z), orientation states bank angle ϕ, pitch attitude angle θ, and heading angle ψ. The camera is modeled with roll, tilt, and pan angles; aspect ratio; zoom; and a variable field-of-view angle. The camera orientation is fixed in the simulations with zero roll; -15 or -20 deg Tilt, depending on the simulation; and -90 deg pan. The definitions of the pan-and-tilt angles are shown in Fig. 4. Note that the angles are negative as shown.

The state transition of the image frame coordinates is calculated as follows. First, the inertial position of the target in the image frame is calculated as

$$\boldsymbol{x}_{\text{im}} = \boldsymbol{x}_t + d(\boldsymbol{x}_c - \boldsymbol{x}_t) \tag{11}$$

where d is defined as in Eq. (10). Here, $\boldsymbol{x}_{\text{im}}$ is the position of the target on the image frame in inertial space; in Fig. 2, $\boldsymbol{x}_{\text{im}}$ is the intersection of the image plane and the line from the image plane to the target in inertial space. This is then converted into the camera frame using a 3-2-1 Euler angle sequence from inertial coordinates to aircraft body coordinates followed by a 3-2-1 Euler angle sequence from the aircraft body-fixed frame to the camera frame. Finally, the image pixel coordinates are calculated using the transformation

$$\begin{Bmatrix} X_{\text{i}} \\ Y_{\text{i}} \end{Bmatrix} = k \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} x_{\text{im,cam}} \\ y_{\text{im,cam}} \\ z_{\text{im,cam}} \end{Bmatrix} \tag{12}$$

where k is a conversion factor from meters to pixels derived from the camera specifications. This transformation matrix maps the three-dimensional (3-D) camera frame, which is defined using body frame conventions, to the two-dimensional image frame.

The kinetic model has steady-state velocity U_1, forces lift L and weight W, and mass m. The aircraft simulation is constructed for planar motion, i.e., flight at constant altitude, constant velocity, and fixed-camera pose. Figure 5 shows the geometry for a steady, level turn. Balancing lift and weight forces in a steady, level $1g$ turn with constant radius R_t and normal acceleration a_n produces the differential equation of motion for heading rate

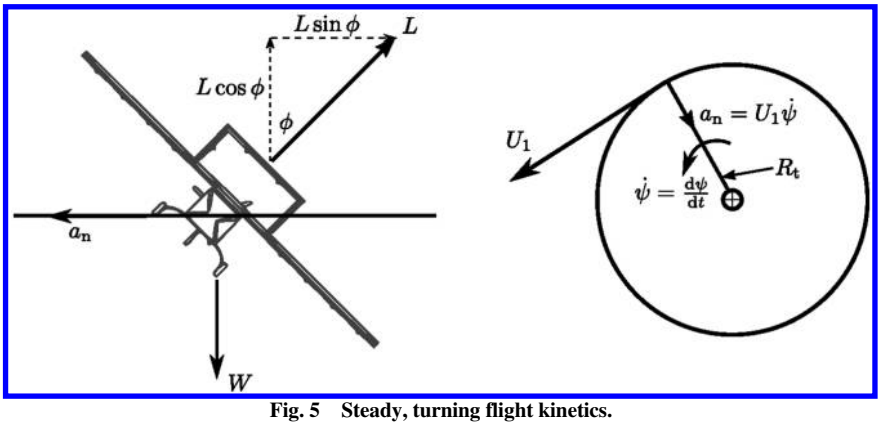

Fig. 5 Steady, turning flight kinetics.

$$
\begin{aligned}
L\cos\phi &= W = mg \\
L\sin\phi &= ma_n = m\dot{\psi}^2 R_t = m\dot{\psi}U_1 \\
\tan\phi &= \frac{m\dot{\psi}U_1}{mg} = \frac{\dot{\psi}U_1}{g} \\
\dot{\psi} &= \frac{g\tan\phi}{U_1} \cong \frac{g}{U_1}\phi
\end{aligned} \tag{13}
$$

The cruise speed and altitude are user-specified variables for the simulation. Equation (13) assumes constant speed, so all simulations presented in this paper will involve constant cruise speed. The actions are increments to the current bank angle ϕ rather than a new total bank angle. The action time span is much greater than the time required to reach a commanded bank angle. The action space of Eq. (4) is used throughout the simulation, but there are limiting factors on the maximum bank angle to the left and right. This is due to both aircraft performance and the image frame of the camera. Because of these limitations, the action space is reduced when a bank limit is reached so that the aircraft can only choose to bank in the opposite direction.

IV. Numerical Results

The RL problem formulated in Sec. II is applied to stationary, linear moving, and randomly moving targets that require different flight paths for tracking. Results for both the Watkins Q-learning algorithm and Watkins $Q(\lambda)$ algorithm are presented. Case 1 addresses stationary targets. Case 2 addresses targets that move in a straight line at constant speed. Case 3 addresses targets that randomly change their heading at constant velocity. Case 4 addresses targets that are being tracked in the presence of wind. The position of the target in the image frame and the time histories of the UAS states are shown for each case described above. All learning takes place via offline simulation and, after a sufficient number of learning episodes have been completed, the control policy performance and robustness are evaluated via Monte Carlo simulation. The Monte Carlo randomization places the initial position of the target in one of the four quadrants of the image frame, and at a random position within each quadrant. The controller must then steer the UAS so that the target remains within the image frame. Image positions are given in pixels, and UAS bank angles are given in degrees. UAS inertial positions are in meters. Monte Carlo simulations are presented for a chosen time span of 500 s. The RL agent was allowed to run for 1,000,000 episodes, with Monte Carlo snapshots taken at a few places beginning at 500,000 episodes. It was determined that more learning following 1,000,000 episodes resulted in diminishing returns.

The results presented here show Monte Carlo simulations of initial condition regions where tracking was possible, and they generally correspond to the simulations that performed the best. Tuning the problem representation to allow for successful learning becomes tractable only when considering a target that is initially within the image frame, since there are initial UAS position conditions either far ahead of the target or far behind the target that prevent the camera from ever seeing the target. Since the UAS is initialized with a zero heading angle, there is also a range of initial conditions for which the UAS is unable to converge to a usable policy due to poor geometry and because there is no target reacquisition.

A. Case 1: Stationary Target

The results shown are taken from one single test case of the Monte Carlo runs in which the target initial conditions place it in the image frame near the center. Figure 6 shows a three-dimensional view of the UAS moving in inertial airspace tracking the target for 500 s. The rectangular projections show the image FOV from the camera on the UAS, with lines from the target position on the ground to the position in the image. Figure 6 shows that the UAS flies a circular orbit to track the stationary target on the ground as expected. Figure 7 shows the position of the target at each time step. The target begins near the origin of the image but does not remain in the center of the image and settles in quadrant 3 for most of the simulation. As the UAS banks left, the target moves up in the image frame, whereas a right bank moves the target down. Time histories of the target position in the image frame and UAS bank angle are shown in Fig. 8, where it is seen that the controller keeps the target in the image frame throughout the 500 s simulation time span.

Figure 9 shows the trajectory of the UAS using the policy learned with the $Q(\lambda)$ algorithm. The UAS attempts to keep the target at the image center but, in doing so, increases the left bank, and thus the heading rate. This causes the UAS to spiral in toward the target and ultimately lose the ability to keep the target in the image frame. Figure 10 shows the location of the target in the image space. The target remains near the center of the X_i axis and in the upper half of the image plane for the majority of the flight until a sharp departure from the image plane at the end. Figure 11

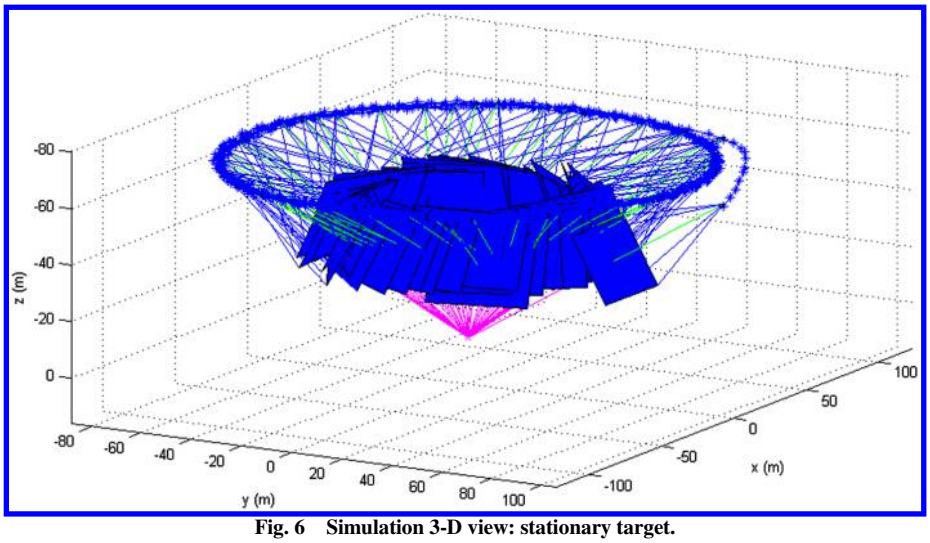

Fig. 6 Simulation 3-D view: stationary target.

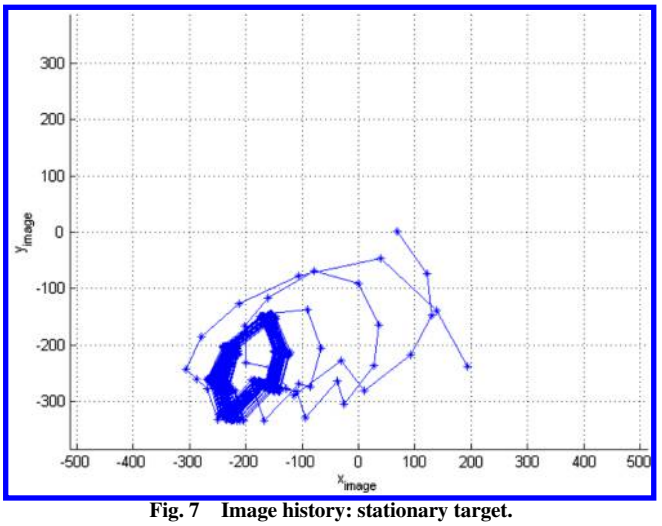

Fig. 7 Image history: stationary target.

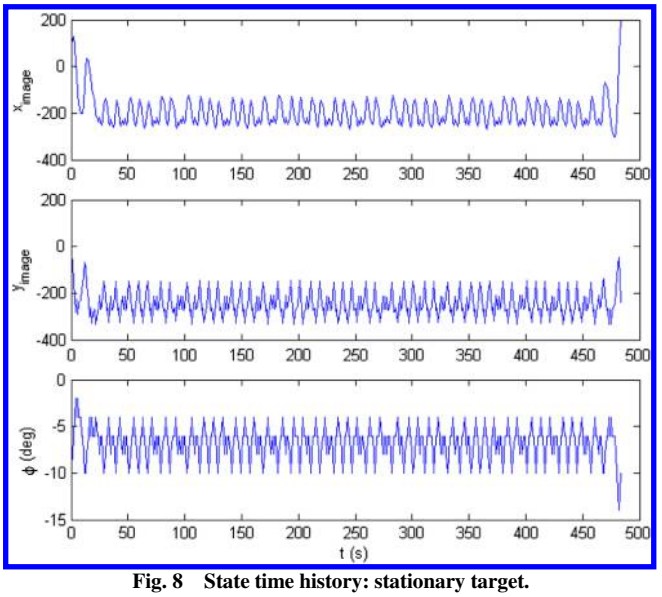

Fig. 8 State time history: stationary target.

presents the time histories for the states, showing that the UAS was able to successfully track the target for almost 200 s. Note the sharp increase in the left bank near the end of the simulation.

B. Case 2: Linear Moving Target

Accounting for target movement becomes a problem using the original state-space representation since the trajectory of the target is not taken into account. Even if it is known that the target is moving in a straight line, the learning algorithm would need some way to handle this movement in the state–action pair decision-making process. To resolve this, the state space is augmented so that first past values are considered as well as current states, as shown in Eq. (14):

$$\boldsymbol{s} = [\,X_{\mathrm{i},k} \quad Y_{\mathrm{i},k} \quad \phi_k \quad X_{\mathrm{i},k-1} \quad Y_{\mathrm{i},k-1} \quad \phi_{k-1}\,]^{\mathsf{T}} \tag{14}$$

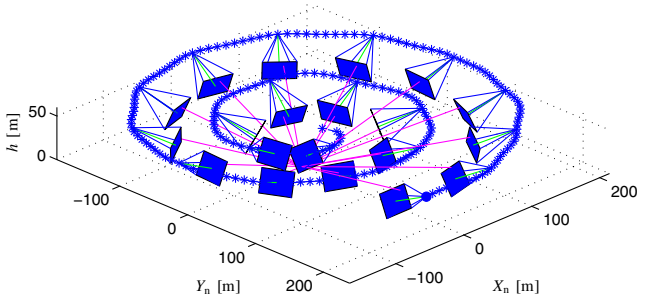

Fig. 9 Simulation 3-D view: stationary target [$Q(\lambda)$ control policy].

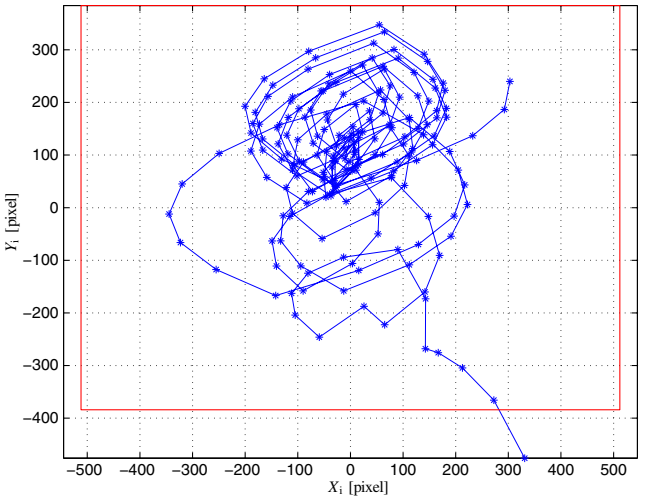

Fig. 10 Image history: stationary target [$Q(\lambda)$ control policy].

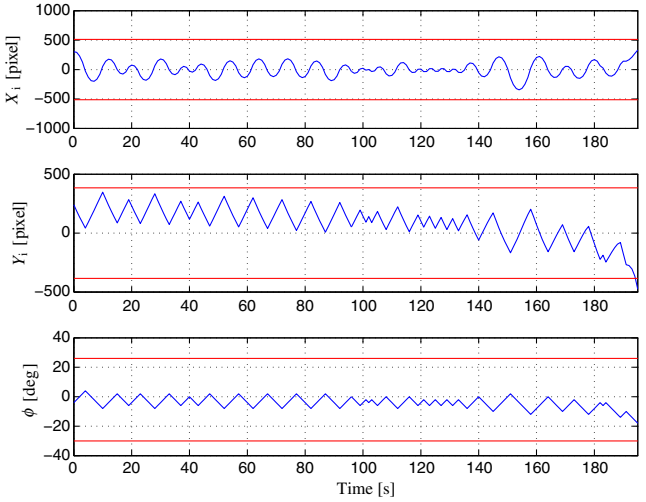

Fig. 11 State time history: stationary target [$Q(\lambda)$ control policy].

with the subscript k denoting the current time index. If the action space and reward structure remain the same as before, then the only alteration made is a dimensional expansion of the action-value function such that the previous time-step states are remembered. This allows the learning agent to account for the fact that the target is moving, since the differences between the first past values of the states and the current states are inherently affected by both the aircraft trajectory and the target trajectory.

With the state space modified for a moving target, new learning was experienced with a target that moves in a straight line at a constant speed equal to that of the UAS. Under this condition, the UAS attempts to fly alongside the target as it travels forward. In Fig. 12, it can be seen that the UAS initially flies alongside the target well and begins to slightly deviate away from it as time progresses. This is due to the stationary camera requiring the tracking to be handled by banking the UAS. It can be seen in Fig. 13 that the target is maintained in the image frame throughout the 500 s duration of the simulation. The time histories shown in Fig. 14 reveal that to maintain this tracking requires frequent modulation of bank angle like the stationary case but, unlike the stationary case, the command frequency slows as the UAS settles into its path alongside the target.

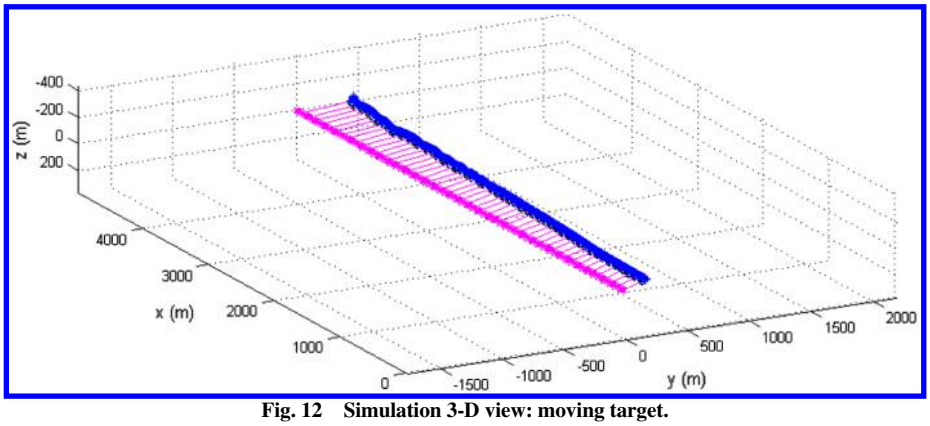

Fig. 12 Simulation 3-D view: moving target.

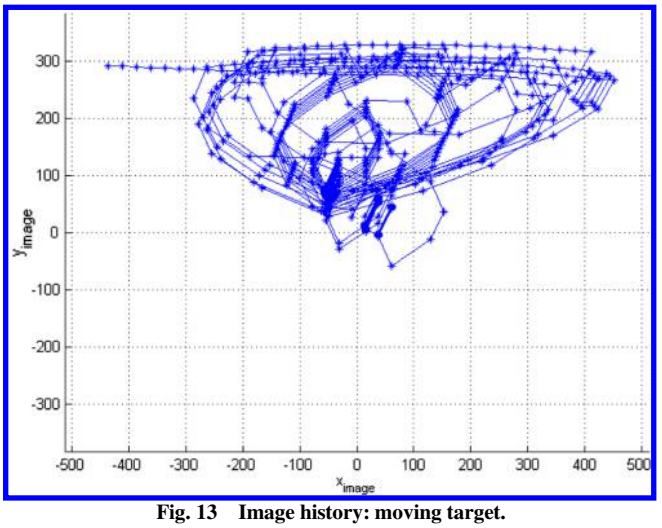

Fig. 13 Image history: moving target.

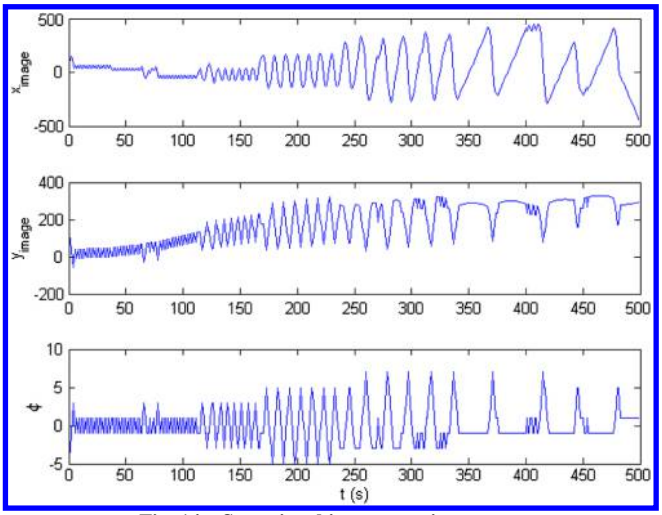

Fig. 14 State time history: moving target.

New learning was also conducted for the $Q(\lambda)$ algorithm and used to track a constant speed target. Figure 15 shows the trajectory the UAS follows while tracking the target. The UAS follows a similar trajectory to that of the Q-learning policy in Fig. 12, maintaining a parallel path to the target. The target is kept in the second quadrant for most of the simulation, after which it moves outside the left image boundary. This occurs due to the UAS yawing to the right and slowly diverging from the path of the target. The target image frame position and UAS state time histories in Figs. 16 and 17 show the UAS keeps the target in the image frame for over 460 s.

V. Target Path Changes

The results presented in Sec. IV demonstrate the capability of this algorithm to learn a control policy for tracking either a stationary target or a linearly moving target with constant speed equivalent to the UAS speed. In real-world ground target tracking scenarios, potential targets do not generally have these simple trajectories. The UAS controller has little or no knowledge of the ground target trajectory and must therefore compensate for unpredictable maneuvering by the target. The learning cannot generalize to all possible target trajectories, but the results in this section will demonstrate that, as long as the variations are not too large, the UAS can track the target using the previously learned stationary target policy. The results presented in this section demonstrate that the policies learned in Sec. IV are robust to small or random changes in target trajectory and that, using this approach, learning separate policies for every type of trajectory is not required.

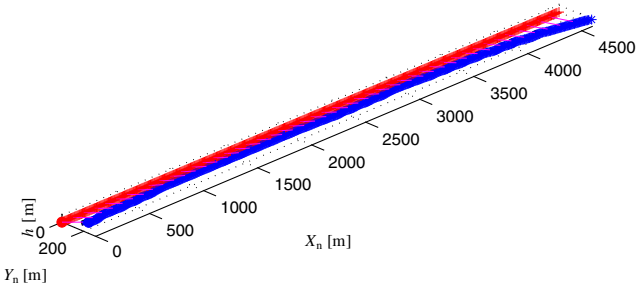

Fig. 15 Simulation 3-D view: moving target [$Q(\lambda)$ control policy].

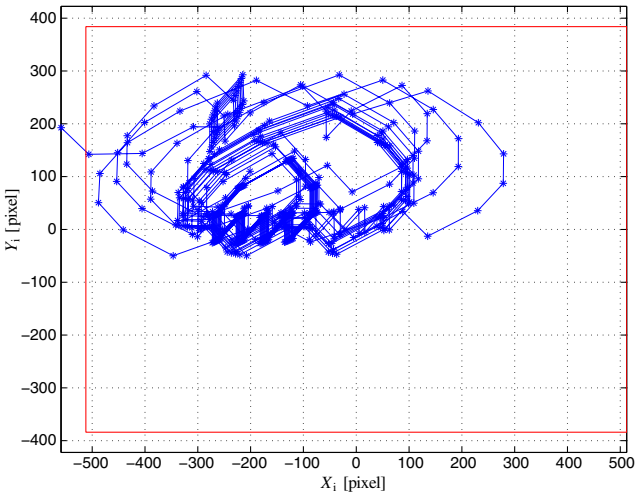

Fig. 16 Image history: moving target [$Q(\lambda)$ control policy].

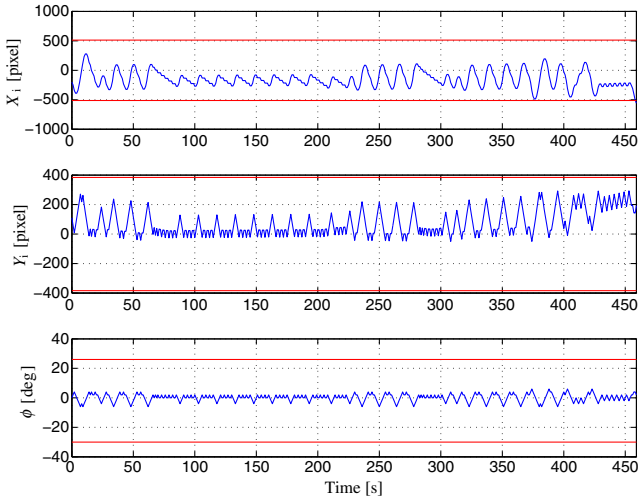

Fig. 17 State time history: moving target [$Q(\lambda)$ control policy].

A. Case 3: Slow Target Movements

When a target is moving too slowly for the UAS to be able to fly directly alongside it, the maneuver must be changed to maintain tracking. Some orbiting of the target is required, but not the same circular orbit that is required in the stationary case. The already learned UAS stationary target control policy from Sec. IV.A can be used to track a target moving at low speed, because approximating it as stationary at any given time step leads to orbiting behavior. As long as the target does not move too rapidly, this approximation works well, since the current position is used for action selection and not the initial position.

The targets considered in these examples move in a straight line at a constant speed. Several target speeds were chosen for testing, and the resulting tracking trajectories are shown in Figs. 18–21. Figures 22–25 present results using the control policy learned with the $Q(\lambda)$ algorithm for the same target speeds.

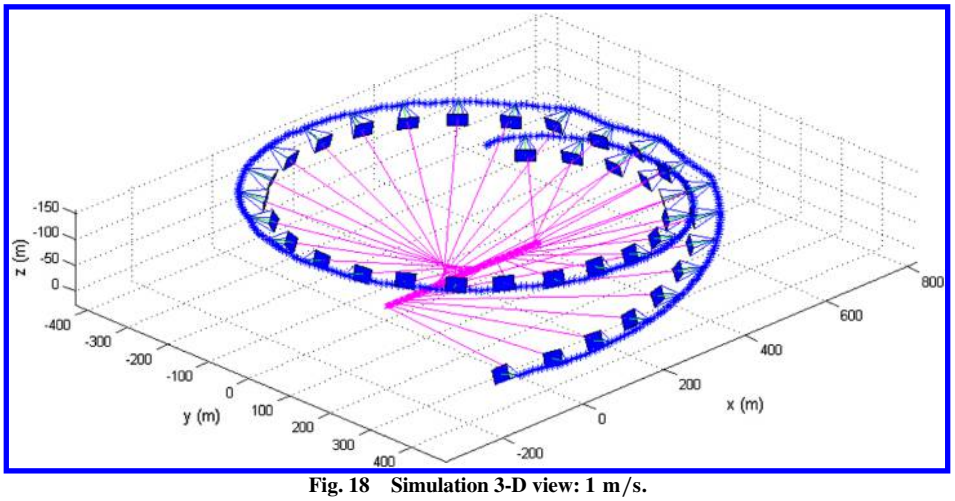

Fig. 18 Simulation 3-D view: 1 m/s.

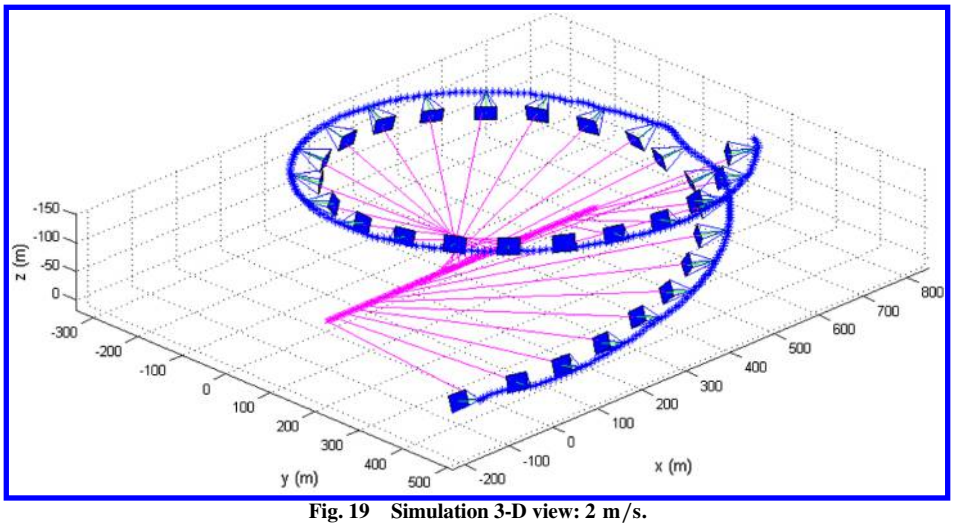

Fig. 19 Simulation 3-D view: 2 m/s.

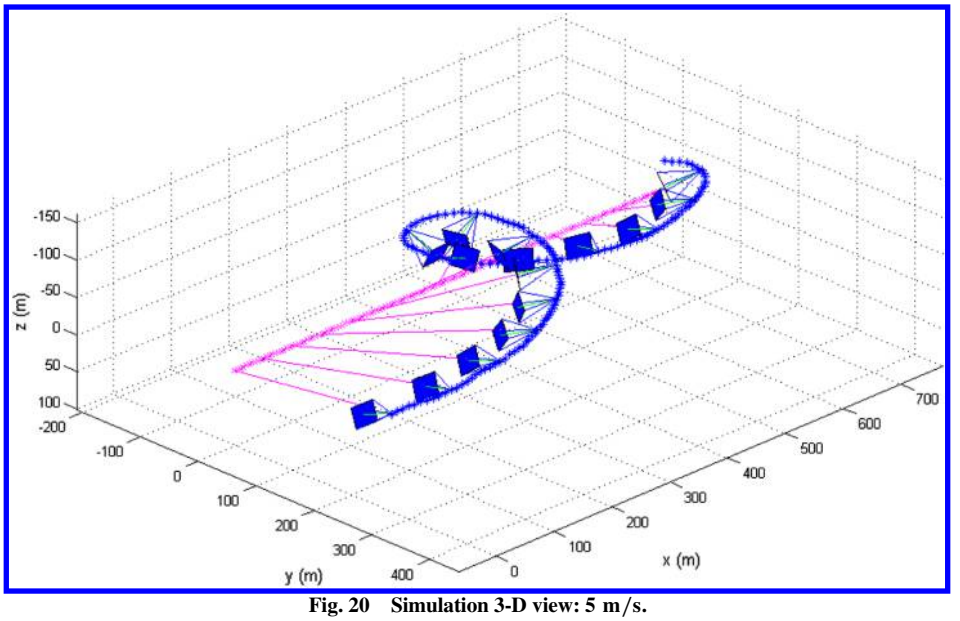

Fig. 20 Simulation 3-D view: 5 m/s.

As the speed of the target increases, the UAS has more difficulty orbiting the target. In Fig. 21, the target is moving too fast, resulting in the UAS losing the target while trying to move into an orbital trajectory. The UAS banks to the left to keep the target in the image frame due to the proximity, but it eventually loses the target out of the bottom of the image frame. However, Figs. 18–20 show that, as long as the target moves relatively slowly, the UAS can track it.

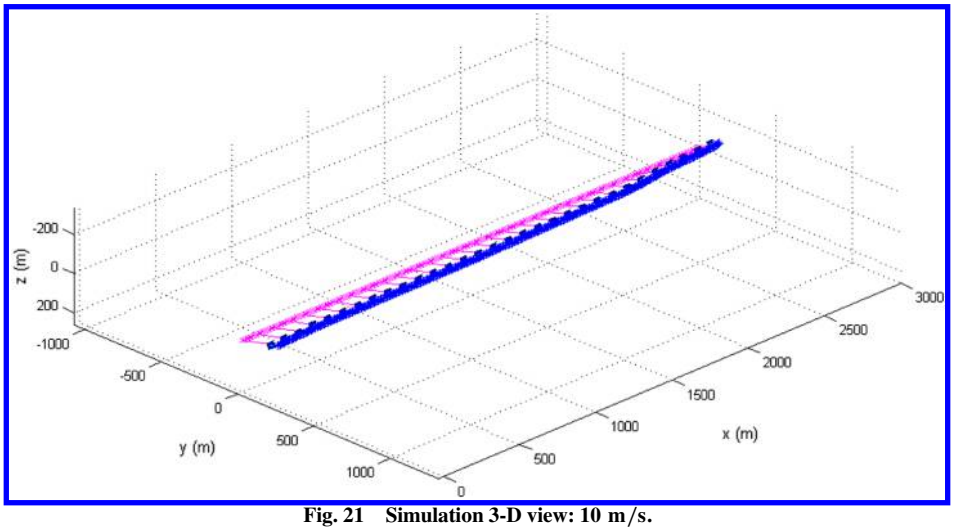

Fig. 21 Simulation 3-D view: 10 m/s.

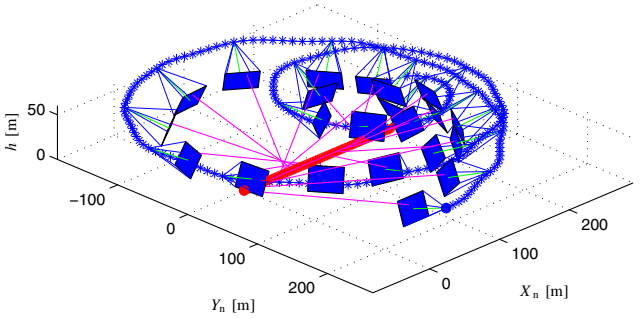

Fig. 22 Simulation 3-D view: 1 m/s [$Q(\lambda)$ control policy].

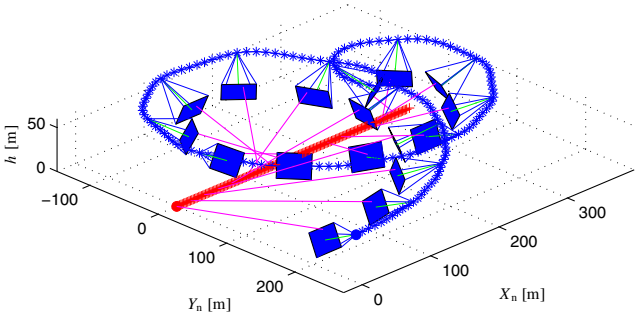

Fig. 23 Simulation 3-D view: 2 m/s [$Q(\lambda)$ control policy].

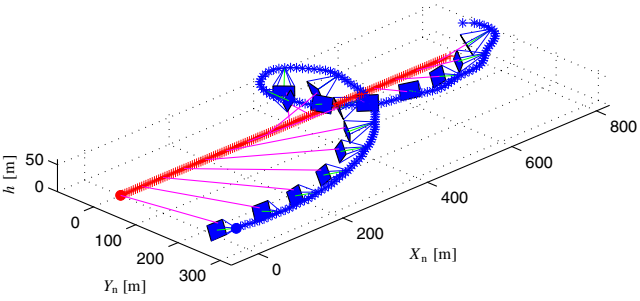

Fig. 24 Simulation 3-D view: 5 m/s [$Q(\lambda)$ control policy].

With eligibility traces, the UAS shows improved performance tracking the 10 m/s target with the stationary policy, and it is able to track the target for approximately 120 s longer. At 360 s into the flight, the UAS starts to diverge from the target path, and the target eventually moves out of the image frame. This policy shows diminished performance at the slower speeds, as the UAS tends to make tight turns and the combination of bank and heading angle move the target out of the frame. However, the UAS tracks the target for at least 180 s for each target speed.

B. Case 4: Target Path Randomization

Targets that are given the same path information to follow but vary their trajectories according to random changes in heading angle are considered. At every time step, the target is given a random angle between $\pm\Delta\psi_{\max}$ that determines the commanded change in heading angle. This simulates a target that chooses to steer randomly within constraints. The resulting update equation for the heading angle at each time step is Eq. (15), where ψ is the heading angle and $\Delta\psi$ is the change in heading angle. Equation (16) shows that the change $\Delta\psi$ is chosen randomly at each time step according to a uniform distribution with a range of $\pm\Delta\psi_{\max}$:

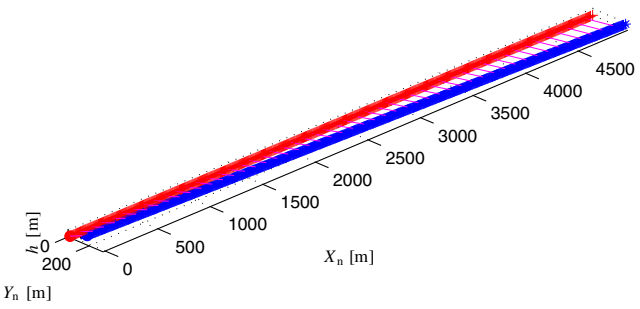

Fig. 25 Simulation 3-D view: 10 m/s [$Q(\lambda)$ control policy].

$$\psi_{k+1} = \psi_k + \Delta\psi_k \tag{15}$$

$$\Delta\psi_k = \mathcal{U}(-\psi_{\max}, \psi_{\max}) \tag{16}$$

With this target path randomization, the control policy that was learned for a moving target in Sec. IV.B was used to track a target with constant speed but an unpredictable path. This was accomplished for a variety of maximum heading angle changes to investigate how the control policy responds to different degrees of path randomization. The five scenarios tested were $\psi_{\max} = 1$, 5, 10, 15, and 45 deg.

Figures 26–30 demonstrate that the same control policy learned for a linear moving target was capable of tracking a target that perturbs its heading from linear trajectory at each time step. All five of these scenarios maintained tracking in the image frame for the full 300 s allotted, demonstrating that the learned policies are capable of tracking stationary and moving targets with path perturbations whether the random deviations are small or large. Again, the success of these policies depends on the initial position of the target relative to the UAS, as there are some initial orientations that do not allow successful tracking due to geometry.

Likewise, Figs. 31–35 present 3-D trajectories of the UAS tracking randomly moving targets using the moving target policy generated with the Watkins $Q(\lambda)$ algorithm. Figure 31 shows similar performance to Fig. 26, maintaining tracking for the full 300 s. Figure 32 shows degraded performance due to the UAS outrunning the target. Still, the UAS is able to autonomously track the target for over 120 s. Figures 33–34 demonstrate a pitfall of using a fixed camera. When the target breaks into the UAS, it is unable to turn to maintain the target in the image frame.

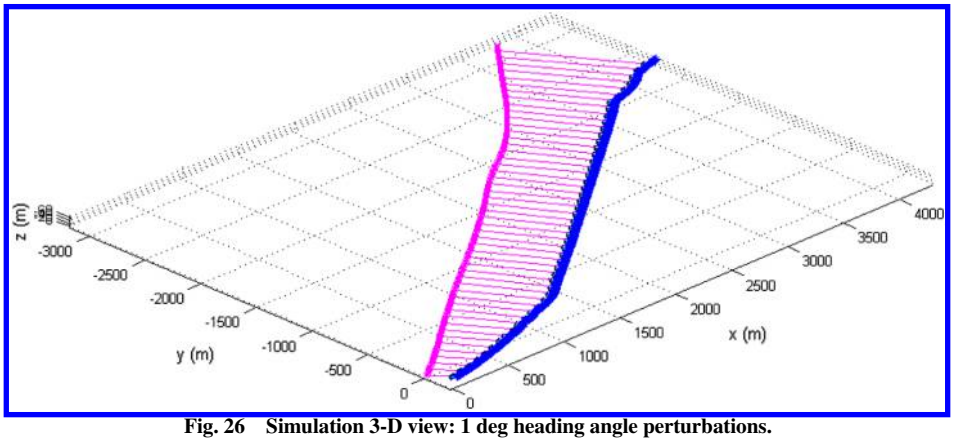

Fig. 26 Simulation 3-D view: 1 deg heading angle perturbations.

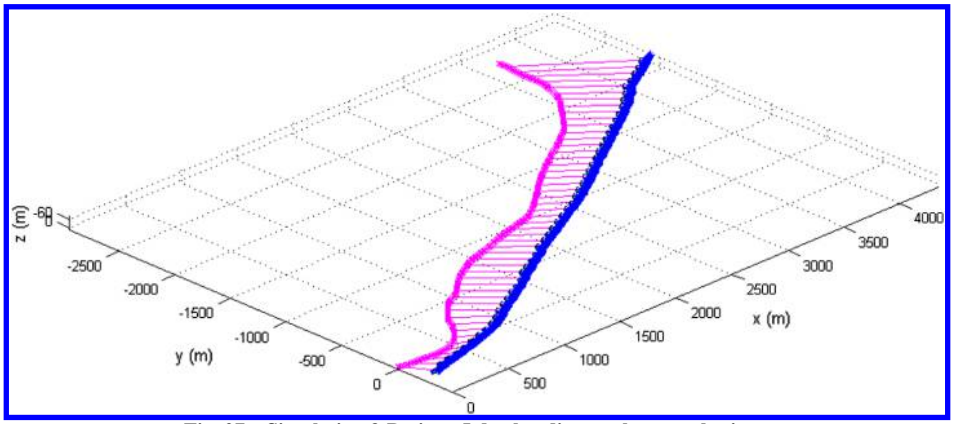

Fig. 27 Simulation 3-D view: 5 deg heading angle perturbations.

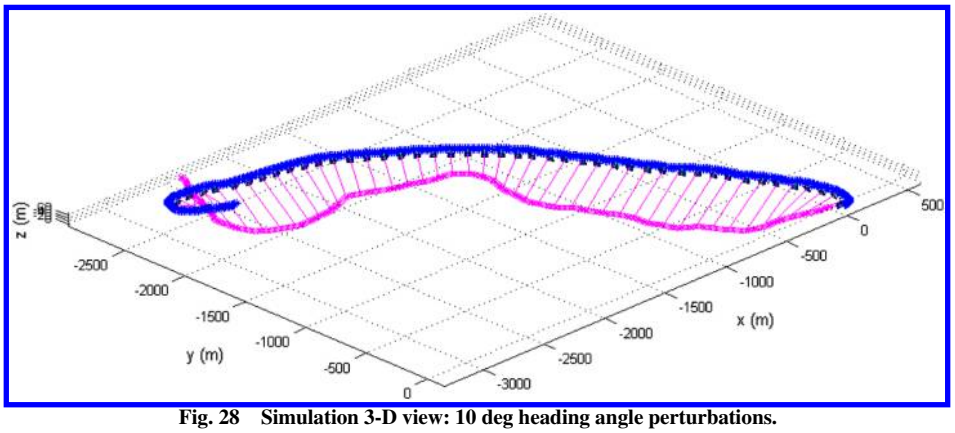

Fig. 28 Simulation 3-D view: 10 deg heading angle perturbations.

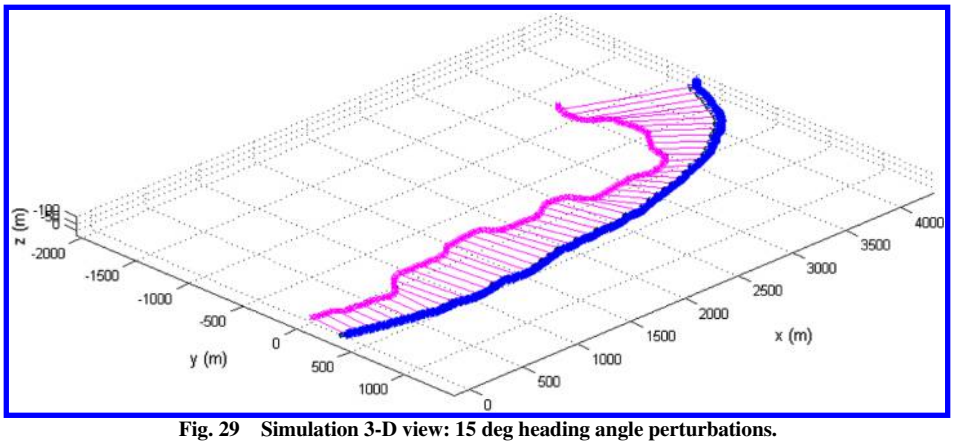

Fig. 29 Simulation 3-D view: 15 deg heading angle perturbations.

This shows that, although the target is unaware of the UAS position and does not evade, an evading target would easily be able to break into the UAS and escape surveillance. Figure 35 again demonstrates the ability of the UAS to track a target with large heading perturbations. In this example, the target turns perpendicular to the UAS flight path, causing it to move out of the image frame, and thereby breaking lock.

VI. Case 4: Wind Disturbance

Sections IV and V presented results with no external disturbances. In this section, wind disturbances are added to the environment. Accounting for wind in the tracking problem has been done using a variety of methods [1,12]. Using the method developed in this paper requires augmenting

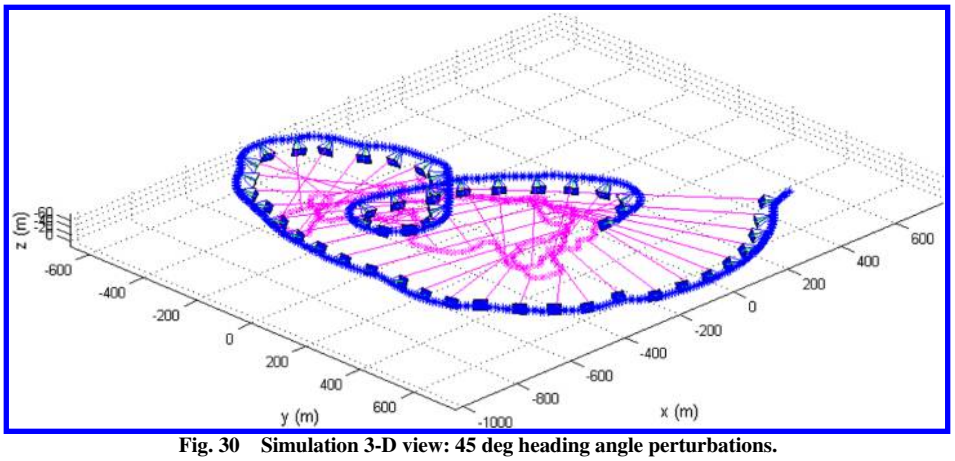

Fig. 30 Simulation 3-D view: 45 deg heading angle perturbations.

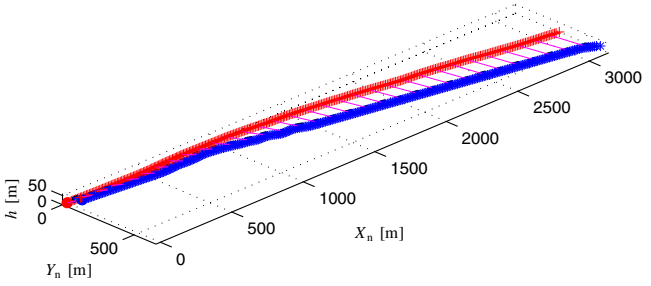

Fig. 31 Simulation 3-D view: 1 deg heading angle perturbations [$Q(\lambda)$ control policy].

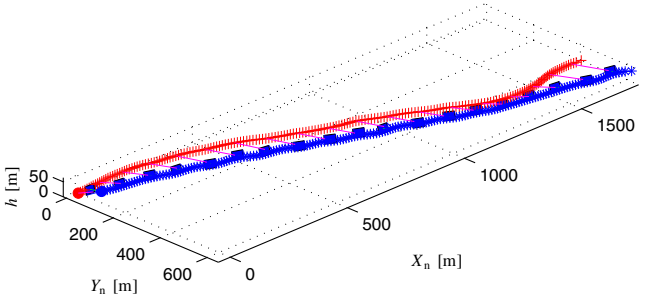

Fig. 32 Simulation 3-D view: 5 deg heading angle perturbations [$Q(\lambda)$ control policy].

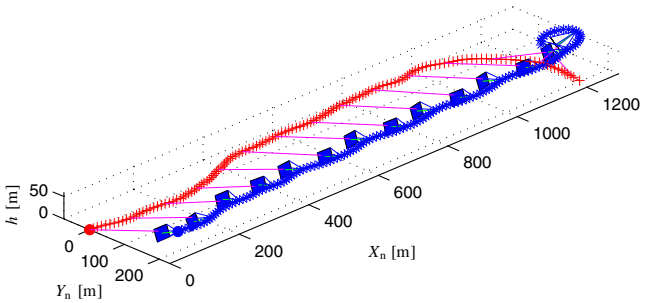

Fig. 33 Simulation 3-D view: 10 deg heading angle perturbations [$Q(\lambda)$ control policy].

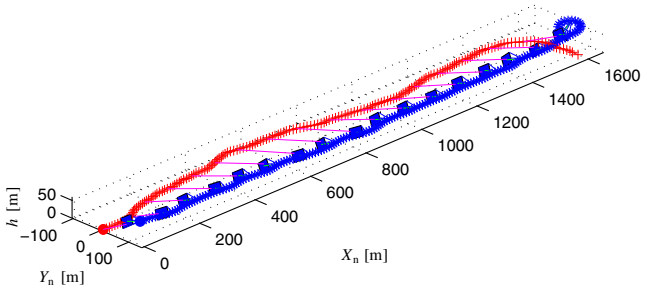

Fig. 34 Simulation 3-D view: 15 deg heading angle perturbations [$Q(\lambda)$ control policy].

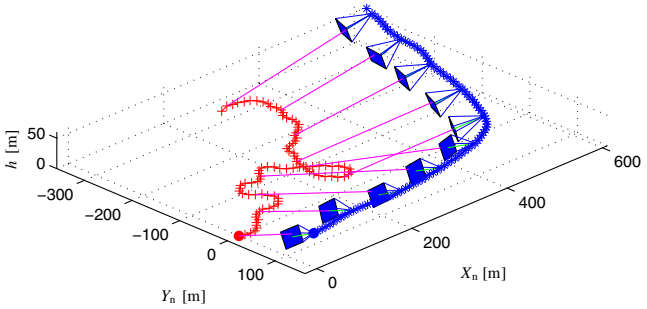

Fig. 35 Simulation 3-D view: 45 deg heading angle perturbations [$Q(\lambda)$ control policy].

the learning state space with two additional states: wind speed and wind direction. This modified state space is shown in Eq. (17), where v_w and ψ_w are the wind speed and wind heading angle, respectively:

$$s = [X \quad Y \quad \phi \quad v_w \quad \psi_w]^{\mathsf{T}} \tag{17}$$

The new learning with wind used random wind speeds and directions initialized at the beginning of each episode. Monte Carlo results were used with the learned Q matrix after 1,000,000 learning episodes. The results of a stationary target with wind disturbance without using eligibility traces

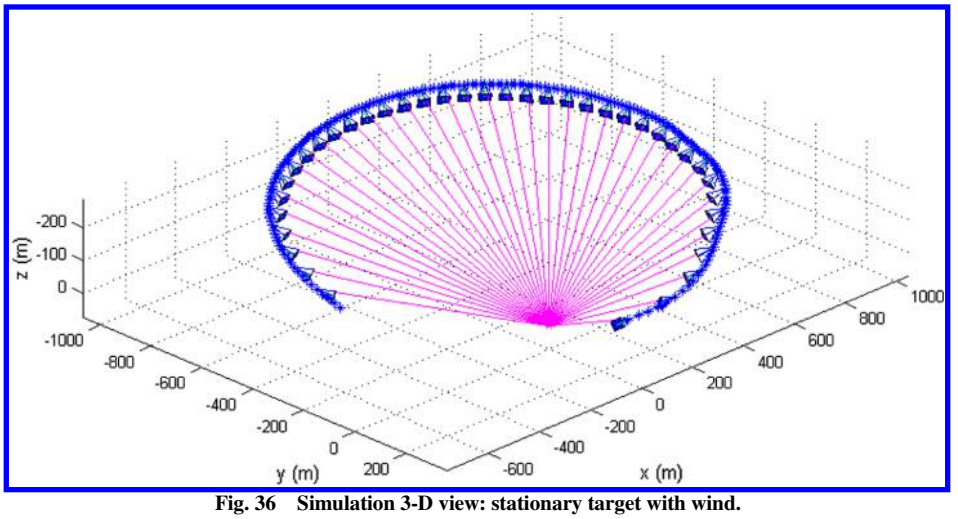

Fig. 36 Simulation 3-D view: stationary target with wind.

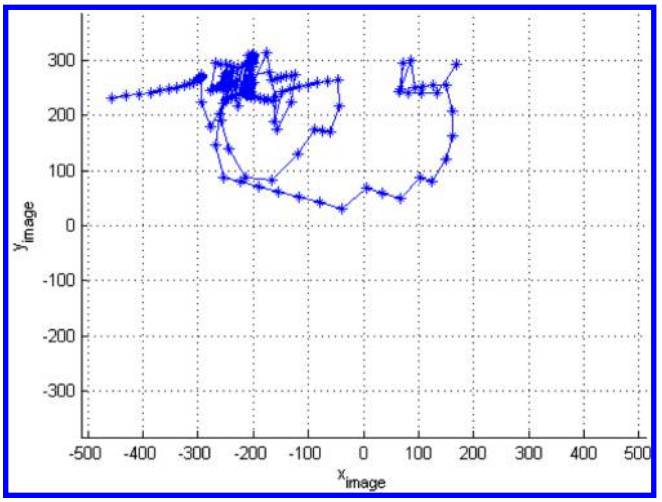

Fig. 37 Image time history: stationary target with wind.

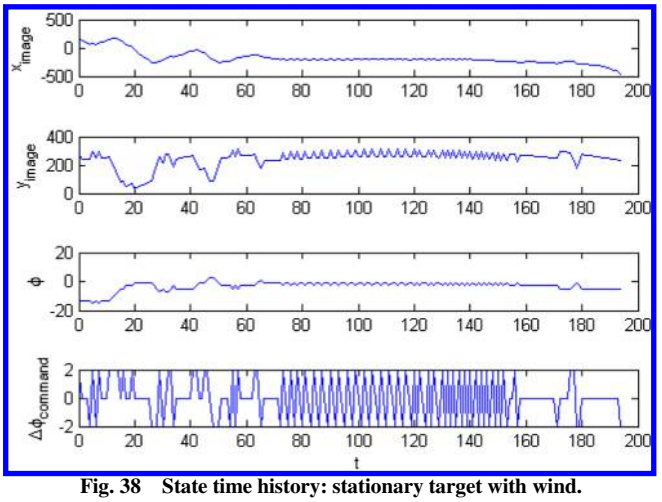

Fig. 38 State time history: stationary target with wind.

are shown. The wind vector for the simulation shown is initialized at 13 mph at a heading angle of 45 deg to the UAS. Figure 36 shows that the UAS flies an elliptical trajectory to account for the presence of the wind, rather than the circular orbit shown in Fig. 6. Figure 37 demonstrates that the target remains in the image throughout the simulation. Figure 38 presents time histories of the target image frame coordinates and the UAS bank angle and bank angle commands.

VII. Conclusions

This paper developed a machine learning algorithm approach for learning control policies that provided visual tracking of stationary and moving ground targets by Unmanned air systems with nongimbaling, fixed pan-and-tilt cameras. Both Q-learning and Q-learning-with-eligibility-traces policies were developed and investigated with a Monte Carlo simulation. Based on the results presented in this paper, the following conclusions were drawn:

1) For stationary targets, the learning algorithm determined the bank angle commands required to enter into and maintain an orbit about the target that kept it in the camera image frame throughout the simulation time span. Although the target did not always stay in the reinforcement learning positive goal area (the origin), the broader tracking goal of keeping the target within the image frame itself for a useful period of time was generally met. For the case of a linear moving target, the learning algorithm determined the bank angle commands that allowed the vehicle to either fly alongside the target or circle the target to keep it in the camera image frame.

2) The learned control policies were robust to small or random changes in target trajectory, so learning separate policies for every type of trajectory was not required. The control policy for a stationary target was suitable without modification for tracking targets that moved at low speed. The control policy for a constant speed, linear motion target was capable of tracking constant speed targets that deviated from their paths by random heading changes. This tracking was possible even for a large degree of randomization in the target path.

3) The learning algorithm can be modified to account for disturbances such as wind that affect the dynamics of the aircraft. By augmenting the learning state space with wind states, a control policy is learned that appropriately alters the aircraft trajectory to maintain tracking. The target still remains in the image frame, but slightly different trajectories are produced.

4) Camera installation and orientation, as well as the initial position of the target relative to the initial position of the aircraft, have a strong influence on the results. With a camera pointing out the left hand side of the aircraft, the controller usually attempts to drive the target into the second quadrant, due to the geometry of the scenario and the location of the camera. Having the target in the second quadrant allows the aircraft to turn ahead of the target to keep it in the image frame in the future. Consequently, for a camera pointing out the right-hand side of the aircraft, the controller drives the target into the first quadrant.

5) Both reinforcement learning algorithms demonstrated successful target tracking for various motion types. Using eligibility traces with the Watkins $Q(\lambda)$ algorithm resulted in good performance with a lower learning cost. Performance could be increased by tuning the hyperparameters

of the learning agent or using a more modern learning agent. Furthermore, Q learning and its tabular representation were simple to implement and use in applications. Ultimately, the learning agent was a component of the overall tracking method and could be replaced as needed with other offpolicy agents.

Acknowledgments

This research is funded by the Raytheon Company, Inc., Intelligence and Information Systems Division, under contract C10-00904, with Technical Monitor Michael R. Moan. It is also supported by the U.S. Air Force Office of Scientific Research under contract FA9550-08-1-0038, with Technical Monitor Fariba Fahroo. This support is gratefully acknowledged by the authors. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Raytheon Company, Inc., or the U.S. Air Force.

References

[1] Rysdyk, R., "Unmanned Air Vehicle Path Following for Target Observation in Wind," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 12, Sept.–Oct. 2006, pp. 1092–1100.
doi:10.2514/1.19101

[2] Rysdyk, R., "UAV Path Following for Constant Line-of-Sight," *2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations Conference, Workshop and Exhibition*, AIAA Paper 2003-6626, Sept. 2003.

[3] Stolle, S., and Rysdyk, R., "Flight Path Following Guidance for Unmanned Air Vehicles with Pan-Tilt Camera for Target Observation," *Proceedings of the 22nd Digital Avionics Systems Conference*, Vol. 2, IEEE Publ., Piscataway, NJ, 2003, pp. 8.B.3-1–8.B.3-12.

[4] Lee, J., Huang, R., Vaughn, A., Xiao, X., Hedrick, J. K., Zennaro, M., and Sengupta, R., "Strategies of Path-Planning for a UAV to Track a Ground Vehicle," *Proceedings of the 2003 Autonomous Intelligent Networked Systems Symposium*, IEEE Publ., Piscataway, NJ, 2003.

[5] Ra, F., Khan, S., Shafiq, K., and Shah, M., "Autonomous Target Following by Unmanned Aerial Vehicles," *Proceedings of the SPIE International Society For Optical Engineering*, Vol. 6230, International Soc. for Optics and Photonics, Bellingham, WA, 2006, pp. 623010-1–623010-8.

[6] Quigley, M., Goodrich, M. A., Griffiths, S., Eldredge, A., and Beard, R. W., "Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera," *ICRA 2005, Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE Publ., Piscataway, NJ, 2005, pp. 2600–2605.

[7] Bethke, B., Valenti, M., and How, J., "Cooperative Vision Based Estimation and Tracking Using Multiple UAVs," *Advances in Cooperative Control and Optimization*, edited by Pardalos, P. M, and Murphey, R., Vol. 369, Lecture Notes in Control and Information Sciences, Springer, Berlin, 2007, pp. 179–189.

[8] Dobrokhodov, V. N., Kaminer, I. I., Jones, K. D., and Ghabcheloo, R., "Vision-Based Tracking and Motion Estimation for Moving Targets Using Small UAVs," *Proceedings of the 2006 American Control Conference*, IEEE Publ., Piscataway, NJ, 2006, pp. 1428–1433.

[9] Redding, J. D., McLain, T. W., Beard, R. W., and Taylor, C. N., "Vision-Based Target Localization from a Fixed-Wing Miniature Air Vehicle," *Proceedings of 2006 American Control Conference*, IEEE Publ., Piscataway, NJ, June 2006, pp. 2862–2867.

[10] Egbert, J., and Beard, R. W., "Low-Altitude Road Following Using Strap-Down EO Cameras on Miniature Air Vehicles," *Proceedings of 2007 American Control Conference*, IEEE Publ., Piscataway, NJ, July 2007, pp. 353–358.

[11] Egbert, J., and Beard, R. W., "Low-Altitude Road Following Using Strap-Down Cameras on Miniature Air Vehicles," *Mechatronics*, Vol. 21, No. 12, 2011, pp. 831–843.
doi:10.1016/j.mechatronics.2010.10.008

[12] Saunders, J., and Beard, R. W., "Visual Tracking in Wind with Field of View Constraints," *International Journal of Micro Air Vehicles*, Vol. 3, No. 3, 2011, pp. 169–182.
doi:10.1260/1756-8293.3.3.169

[13] Theodorakopoulos, P., and Lacroix, S., "A Strategy for Tracking a Ground Target with a UAV," *IEEE Conference on Intelligent Robots and Systems*, IEEE Publ., Piscataway, NJ, Sept. 2008, pp. 7–9.

[14] Sutton, R. S., "Dyna: An Integrated Architecture for Learning, Planning, and Reacting," *ACM SIGART Bulletin*, Vol. 2, No. 4, 1991, pp. 160–163.
doi:10.1145/122344

[15] Moore, A. W., and Atkeson, C. G., "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time," *Machine Learning*, Vol. 13, No. 1, 1993, pp. 103–130.
doi:10.1007/BF00993104

[16] Lagoudakis, M. G., and Parr, R., "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, Vol. 4, Dec. 2003, pp. 1107–1149.

[17] Barto, A. G., and Mahadevan, S., "Recent Advances in Hierarchical Reinforcement Learning," *Discrete Event Dynamic Systems*, Vol. 13, Nos. 1–2, 2003, pp. 41–77.
doi:10.1023/A:1022140919877

[18] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237–285.
doi:10.1613/jair.301

[19] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction, Adaptive Computation, and Machine Learning*, MIT Press, Cambridge, MA, 1998.

[20] Watkins, C. J. C. H., and Dayan, P., "Q-Learning," *Machine Learning*, Vol. 8, 1992, pp. 279–292.

[21] Watkins, C. J. C. H., "Learning From Delayed Rewards," Ph.D. Dissertation, Univ. of Cambridge, Cambridge, England, U.K., 1989.

J. P. How
Associate Editor